REMARKS

Claim 1 is amended to remove the objected to language of "without special hardware support or special loop control instruction. " The word "software" is added. The reason for rejecting the claims under 35 U.S.C. 112 is therefore overcome.

Claims 1-4 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rau et al " Code Generation Schema for Modulo Scheduled Loops", ACM Proceedings of the 25th annual International Symposium on Microarchitecture, Dec. 1992, volume 23, iss. 1-29 hereinafter Rau_1 in view of Rau et al., "Register Allocation for Software Pipelines Loops" , June 1992, InProc. Of the ACM SIGPLAN'92 Conference on Programming Language Design and Implementation, pages 283-299 (hereinafter Rau_2), and further in view of Akkary, USPN 6,240,509 (hereinafter Akkary) and Bringmann, "Enhancing Instruction Level Parallelism through compiler-controlled speculation", Univ. of Illinois, 1995 (hereinafter Bringmann).

In Rau_1, the authors do claim that an irregular loop can be software pipelined if the entire epilog can be collapsed. Obvious or not, we cannot claim this as new, but this is not the contribution of our invention.

The primary challenge of collapsing the epilog for an irregular loop is two fold: recurrence constraints and register pressure (esp. prediate pressure). Recurrence constraints (aka recurrence cycle, dependence cycle - should be defined in patent) refer to the order in which operations must execute. In processing the irregular loop so that the epilog can be collapsed, it is a challenge to minimize the number of new instructions added on the critical cycle. Lengthening this critical cycle slows down the loop by slowing down the rate at which new iterations can be initiated.

The contribution of applicant's invention is software pipelining irregular loops *without* the special purpose hardware and with a limited NON-rotating register file, and a very small predicate register file (5 or 6 predicate registers). Secondarily, there are a limited number of slots and we MUST avoid drastic increases in code size (e.g., multiple epilogs).

Although it might be relatively obvious how to do this with an unlimited static (i.e., non-rotating) register file and predicate register file, it is NOT obvious that this can be done effectively in practice with a small register file and predicate register file. In fact, the first version of applicant's paper was rejected, because the paper reviewers were skeptical that our method would work.

The minimum hardware that Rau_1 relies on is hardware for speculative execution. This is a non-significant, complex piece of hardware that is not realistic for all processors, especially many in the embedded world for a variety of reasons, including cost and power. These are not issues in the Rau world (general purpose processors). Their alternate hardware configuration has rotating register and hardware for speculative code motion.

Rau_2 depends on dynamic registers (aka rotating register files) or MVE (software-only approach) to software pipeline irregular loops. MVE has a big drawback: code size!! With MVE, loop unrolling (making multiple copies of a loop body and multiple epilogs are *required*, both *very* expensive in terms of code size. Additionally, loop unrolling often increases register pressure significantly (from our

experience - mileage may vary) which is problematic for loops that are already register-pressure bound.

Akkary

Akkary relies on trace buffers for speculative execution, again a non-trivial piece of hardware. Again, it's a different ballgame when you can put this type of hardware into your processor.

Bringmann

The Bringmann thesis discusses compiler techniques implemented within the framework of the IMPACT compiler. Chapter 3 describes the concept of compiler-controlled speculation. Examples of speculation models are described for different speculation classes. Write-back suppression is presented in Chapter 4 to present an alternative compiler-controlled speculation model that requires some processor assistance to perform recovery. Chapter 5 describes an alternative speculation model that can be used to enhance the performance of existing speculation models. The examiner makes reference to page 52, chapter 4.3.2. on register allocation extension. The register allocator assumes that all locatable operands reside within virtual registers. For each of these virtual registers it constructs a live range which consists of the set of instructions where the operand is alive. Allocation then proceeds by coloring an interference graph constructed from these live ranges. It is not seen where this teaches applicants claimed invention.

Applicant's claim 1 call for: "A method of software pipelining program loops having irregular loop control comprises the steps of:

determining which instructions in loop code in a memory may be speculatively executed,

storing in a computer memory a set of registers that are modified by an instruction and are alive out of the loop, and

modifying the program code so that the values of those registers are saved to a temporary register during all proper iterations, and

copying back to a register the value of the temporary register once the loop is completed."

This is not taught or suggested by the references. The examiner argues the combination of Rau et al " Code Generation Schema for Modulo Scheduled Loops", ACM Proceedings of the 25th annual International Symposium on Microarchitecture, Dec. 1992, volume 23, iss. 1-29 hereinafter Rau_1 in view of Rau et al., "Register Allocation for Software Pipelines Loops" , June 1992, InProc. Of the ACM SIGPLAN'92 Conference on Programming Language Design and Implementation, pages 283-299 (hereinafter Rau_2), and further in view of Akkary, USPN 6,240,509 (hereinafter Akkary) and Bringmann, "Enhancing Instruction Level Parallelism through compiler-controlled speculation", Univ. of Illinois, 1995 (hereinafter Bringmann). Nowhere in these reference is such a combination taught in these references. The examiner argues that in a method using speculation in handling pipelined loops analogous to that of Rau_1 Bringmann discloses register extensions and Akkay discloses the use of temporary registers and instruction trace buffer for extension of the speculation- intensive pipelined in order to prevent mis-speculation recovery resources. The examiner then concludes that this teaches the committing or copying of values to and from secondary registers to provide for mishaps recovery and roll back situations from secondary registers to provide for mishaps recovery and rollback situations from the course of taking a wrong path

during execution. The examiner argues that when register resources are available, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement to 2-copies techniques with hardware support by Rau_2) the use of register support such as register extension by Bringmann, or temporary registers as taught by Akkay, because this would allow extended means to provide for mishaps recovery and rollback situations so to make optimal use of architectural resources as intended by both Rau_1 and Rau_2. This whole argument is not based on the teaching of the references themselves but on an examiners hindsight reconstruction using references that do not point to such a combination. The explanation and the number of references involved coupled by the examiners need to rely on his speculative combinations makes it clear that the subject invention is not obvious in view of the references.

Furthermore, in regard to combining the cited prior art, reference is made to In re Fritch, 23 USPQ2d 1780 and particularly the portion thereof at page 1783 under "Prima Facie Obviousness" where the Court stated:

"In proceedings before the Patent and Trademark Office, the Examiner bears the burden of establishing a prima facie case of obviousness based upon the prior art. '[The Examiner] can satisfy this burden only by showing some objective teaching in the prior art or that knowledge generally available to one of ordinary skill in the art would lead that individual to combine the relevant teachings of the references.' The patent applicant may then attack the Examiner's prima facie determination as improperly made out, or the applicant may present objective evidence tending to support a conclusion of nonobviousness.'"

and later stated:

"'Obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination. Under section 103, teachings of references can be combined only if there is some suggestion or incentive to do so.' Although couched in terms of combining teachings found in the prior art, the same inquiry must be carried out in the context of a purported obvious 'modification' of the prior art. The mere fact that the prior art may be modified in the manner suggested by the Examiner does not make the modification obvious unless the prior art suggested the desirability of the modification."

There is no suggestion in the references to suggest the combination claimed or the desirability of the modification.

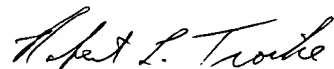Claim 1 is therefore deemed allowable over the references.

Claims 2 and 3 dependent on claim 1 is deemed allowable for at least the same reasons as Claim 1.

Claim 4 is deemed allowable for the same reasons as Claim 1.

In view of the above applicants Claims 1-4 are deemed allowable and an early notice of allowance of these claims is deemed in order and is respectfully requested.

If the examiner persists in the rejection of the claims applicant respectfully requests the amendment be entered to limit the issues on appeal.

Respectfully requested;

Robert L. Troike (Reg. 24183)

Telephone No.(301) 259-2089